Fast Haptic Rendering of Complex Objects Using Subdivision Surfaces

Chris Raymaekers Koen Beets Frank Van Reeth

Expertise Centre for Digital Media, Limburg University Centre
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{*chris.raymaekers, koen.beets, frank.vanreeth*}*@luc.ac.be*

Abstract

Haptic rendering of meshes of arbitrary topology is a difficult and time-consuming process. This paper presents the use of subdivision surfaces in order to solve this problem. An overview of subdivision surfaces is given and the algorithms needed to calculate the surface contact point are discussed. We will show that this algorithm is faster than traditional algorithms.

Introduction and Related Work

Haptic rendering of complex objects needs a lot of calculating power. Even with modern computers, limiting the number of calculations that need to be made on a complex object can be very difficult. Two approaches are frequently used: mathematical representations of the objects, and polygonal models. The GHOST SDK (SensAble, 2001) uses both methods: a mathematical representation is used for simple shapes, such as cubes and cones, while arbitrary meshes are represented by a polygonal model.

Most of the time, only a limited number of polygons is used in order to keep the calculations within the 1ms interval of the haptics loop. However, this also limits the precision of the model. An alternative is the use of mathematical representations, such as NURBS. This however introduces other problems: representing models of arbitrary topology is for instance very difficult.

In our research, we would also like to deform the objects. Using NURBS, the seams of the patchwork can become visible during deformation. This problem also occurs in computer animation and has been solved by using subdivision surfaces. A famous example is the Pixar movie "Geri's Game" (DeRose et al., 1999).

Subdivision Surfaces

A subdivision surface is a surface that is defined as the limit of a series of refinements $M_1, M_2, \ldots$, starting from an original control mesh M_0. Because of this property, they support level-of-detail. Since they can also be used to efficiently represent objects of arbitrary topology, and they can be modified easily to support features such as creases and boundaries, it is no surprise subdivision surfaces are already used in computer graphics and computer animation. Figure 1 shows a control mesh and the first refinement.

A wide variety of subdivision schemes for surfaces exist, with an equally large variety in properties. Two of the most well-known subdivision schemes for surfaces are the Catmull-Clark scheme, which works on quadrilateral meshes, as described in (Catmull and Clark, 1978), and the Loop scheme, which is triangular (Loop, 1987). The types of surfaces generated by these schemes differ, but the general principles of subdivision surfaces remain the same. In our research, we use the triangular loop scheme because triangular polygons are very well suited for modelling freeform surfaces, and they can be easily connected to an arbitrary configuration . For a detailed explanation of subdivision and subdivision surfaces, we refer the interested reader to (Zorin and Schröder, 2000).

The Loop scheme, based on the three-dimensional box spline, is an approximating face-split scheme for triangular meshes, invented by Charles Loop. The resulting surfaces are C^2 everywhere except at extraordinary vertices — with a valence different from 6 — where they are C^1.

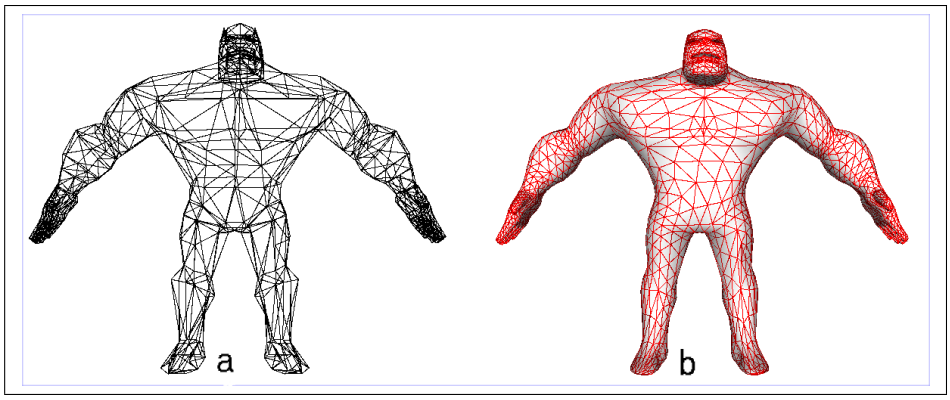

Figure 1: Loop control mesh and first refinement

An iteration of the scheme consists of two stages. In the first stage, known as the splitting stage, a new vertex is added in the middle of each edge, and both the old and new vertices are connected to form 4 new triangles for each old triangle. In the smoothing stage, all vertices are averaged with their surrounding vertices. This smoothing step, together with the weights used, is visualized in figure 2. Loop's original choice for β was $\beta = \frac{1}{k}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}cos(\frac{2\pi}{k})^2)$, where k is the valence of the central vertex, but other choices are possible as well (Warren, 1995). Figure 2 also shows that the support — which is the region over which a point influences the shape of the limit surface — of the Loop scheme is small and limited.

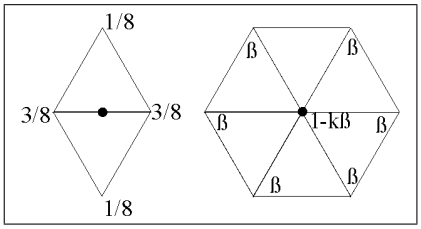

Figure 2: Subdivision Mask of the Loop Scheme

Calculations using Loop Surfaces

During haptic rendering of polygon meshes, all of the object's faces generally need to be processed. This is no longer necessary with subdivision surfaces. The multiresolution properties of subdivision surfaces can be exploited so that only the control mesh has to be processed as a whole. In the processing stages of the following, more detailed, subdivision levels, the results of the previous test are used, thus leading to a smaller number of polygons that have to be processed. This leads to a huge increase in speed.

As mentioned in the previous section, a subdivision surface is defined as the limit of a series of surfaces. This gives rise to two interesting properties:

- Every face at level $n-1$ can be linked to four faces at subdivision level n.
- As can be seen from figure 2, the new co-ordinates of a vertex are influenced by the surrounding vertices. Using the loop subdivision scheme, most vertices have a valence of 6, so 6 vertices are needed to calculate the new co-ordinates. Generalizing this for a triangle (figure 3), each vertex of the triangle is influenced by its 6 neighboring triangles. Since a number of these neighboring triangles are shared, 12 neighboring triangles are needed to calculate the new co-ordinates of each of the triangle's vertices. The grey triangle in figure 3 is the triangle that is subdivided.

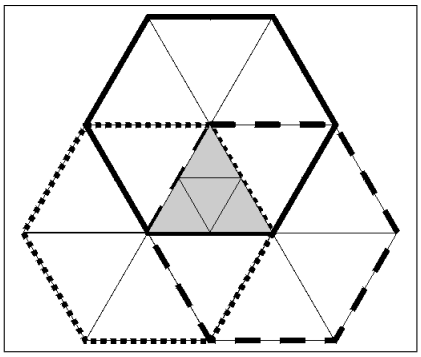

Figure 3: Area which influences a single triangle

The next two sections explain the 2 problems that need to be solved. In both cases the algorithm starts from the control mesh at level 0, leading to an accuracy up to an arbitrary subdivision level n, which contains 4^n times as much triangles as level 0 does.

Performing the inside-outside test

Consider a control mesh M_0, consisting of a triangles. The following steps describe the algorithm that checks wether a point p lies inside or outside the object.

1. Shoot a semi-infinite ray, starting at point p.
2. Select all the intersected faces and the face closest to the point being tested.
3. Extend this selection by including all triangles in the 1-neighborhoods of all vertices of the intersected faces.
4. Replace all the selected faces by their subdivided children. The number of triangles is multiplied by 4. Again test all triangles in this selection for intersection with the ray.
5. Check wether the number of intersections found is different from the previous number. This can happen because the refined meshes "shrink" in areas where the control mesh is convex. In concave regions, the refined meshes grow outside of the control mesh.

   If the number has changed, go to step 7, otherwise proceed with step 6.
6. If the required subdivision level is not yet reached, go to step 2.
7. If the number of intersections is odd, the point lies inside the polymesh. Otherwise it lies outside the polymesh.

Calculating the SCP

Using the results of the previous calculations, the SCP can be calculated.

1. If in the previous algorithm the required subdivion level is found go directly to step 7.
2. Select those faces from the selected faces of the current subdivision level which are closest to the point being tested.
3. Extend the selection to faces which contain vertices in the 1-neighborhood of all vertices in the selected faces.
4. Replace the selection with its next subdivision level.

5. Select the closest face from the previous selection.
6. If the required subdivision level is not yet reached, go to step 3.
7. Calculate the exact intersection point of the closest face. This is the SCP.
8. For each vertex of the triangle the limit normal vector is calculated by taking the vector product of the following two vectors.

$$t_1 = \sum_{i=0}^{k-1} \cos\frac{2\pi i}{k} p_i \qquad t_2 = \sum_{i=0}^{k-1} \sin\frac{2\pi i}{k} p_i$$

The normal in the SCP is calculated by interpolating these three normals.

Comparison

Using subdivision surfaces, only a small number of triangles need to be processed. Consider again the control mesh M_0, consisting of a triangles. At level 0 a intersection tests have to be performed. If k intersections are found (k is typically a very small number), in each step these triangles and their neighboring triangles need to be subdivided. In the worst case scenario, where the neighboring zones are al disjunct, $k*(1+12)*4 = k*52$ intersection tests have to be performed for each subdivision level. The maximum number of tests (if the test is not successful before reaching level n and all zones are always disjunct) is $a + n * k * 52$ (please note that the subdivision process is performed in preprocessing stage) . If a "normal" polymesh with the same number of polygons has to be checked, $a * 4^n$ triangles would have to be checked.

For instance, suppose a control mesh, consisting of 100 triangles, is checked until level 4 (a typical level) and 5 intersections are found. This leads to $100 + 4 * 5 * 52 = 1140$ checks. The equivalent polymesh consists of $a * 4^4 = 25600$ triangles which leads to more than 20 times as much checks.

When using adaptive subdivision, where the fact if a triangle is subdivided depends on the surface area of the triangle (compared to the other triangles of the mesh), even a smaller number of tests is needed.

Conclusions

We proposed a technique for fast haptic rendering by using subdivision surfaces, which can greatly improve speed. Also, by using subdivision surfaces, we can evaluate objects of any topology up to an arbitrary refinement level.

Acknowledgments

Part of the work presented in this paper has been funded by the Flemish Government and EFRO (European Fund for Regional Development).

References

Catmull, E. and Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *CAD*, 10(6):350–355.

DeRose, T., Kass, M., and Truong, T. (1999). Subdivision surfaces in character animation. In *Proceedings of the SIGGRAPH 1999 annual conference on Computer graphics*, pages 85–94, Los Angeles, CA, USA.

Loop, C. (1987). Smooth subdivision surfaces based on triangles. Department of mathematics, University of Utah, Utah, USA.

SensAble (1996–2001). *GHOST Programmers Guide*.

Warren, J. (1995). Subdivision methods for geometric design. Rice University.

Zorin, D. and Schröder, P. (2000). *Subdivision for Modeling and Animation*. Number 23 in Course Notes for SIGGRAPH 2000. ACM.